Optymalizacja zapytań w środowisku heterogenicznym CPU/GPU dla baz danych szeregów czasowych

Autoreferat rozprawy doktorskiej

Piotr Przymus

19 marca 2014

1 Wstęp

W ostatnich latach przetwarzanie i badanie szeregów czasowych zyskało spore zainteresowanie. Rosnące ilości danych i potrzeba ich sprawnego przetwarzania nadały nowe kierunki prowadzonym badaniom, które uwzględniają również wykorzystanie rozwiązań sprzętowych.

Procesory graficzne (GPU) mają znacznie więcej zastosowań niż tylko wyświetlanie obrazów. Coraz częściej są wykorzystywane przy rozwiązywaniu problemów obliczeniowych ogólnego zastosowania, które mogą spożytkować możliwości przetwarzania masywnie równoległego. Wiele źródeł potwierdza skuteczność GPU zarówno w nauce, jak i w zastosowaniach w przemyśle. Jest jednak kilka kwestii związanych z użyciem GPU jako koprocesora w bazach danych, które trzeba mieć na uwadze.

Po pierwsze, wszystkie obliczenia na GPU są poprzedzone czasochłonnym transferem danych. W pracy zaprezentowano rezultaty badań dotyczących lekkich i bezstratnych algorytmów kompresji w kontekście obliczeń GPU i systemów baz danych dla szeregów czasowych. Omówione zostały algorytmy, ich zastosowanie oraz szczegóły implementacyjne na GPU. Rozważono wpływ algorytmów na wydajność przetwarzania danych z uwzględnieniem czasu transferu i dekompresji danych. Ponadto, zaproponowany został adaptacyjny planer kompresji danych, który wykorzystuje różne algorytmy lekkiej kompresji w celu dalszego zmniejszenia rozmiaru skompresowanych danych.

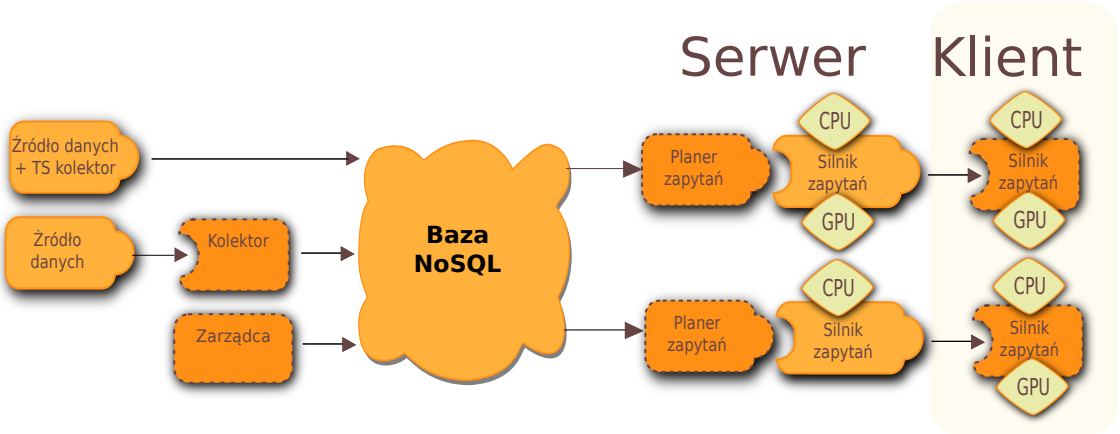
Kolejnym problemem są zadania, które źle (lub tylko częściowo) wpisują się w architekturę GPU. Może być to związane z rozmiarem lub rodzajem zadania. W pracy zaproponowany został model heterogenicznych obliczeń na CPU/GPU. Przedstawiono metody optymalizacji, poszukujące równowagi między różnymi platformami obliczeniowymi. Opierają się one na heurystycznym poszukiwaniu planów wykonania uwzględniających wiele celów optymalizacyjnych. Model leżący u podstaw tego podejścia naśladuje

rynki towarowe, gdzie urządzenia są traktowane jako producenci, konsumentami są natomiast plany zapytań. Wartość zasobów urządzeń komputerowych jest kontrolowana przez prawa popytu i podaży. Zastosowanie różnych kryteriów optymalizacji pozwala rozwiązać problemy z zakresu heterogenicznego przetwarzania zapytań, dla których dotychczasowe metody były nieskuteczne. Ponadto proponowane rozwiązania wyróżnia mniejsza złożoność czasowa i lepsza dokładność.

W rozprawie omówiono przykładowe zastosowanie baz danych szeregów czasowych: analizę zachowań racicznicy zmiennej (*Dreissena polymorpha*) opartą na obserwacji rozchyleń muszli zapisanej w postaci szeregów czasowych. Proponowany jest nowy algorytm oparty na falkach i funkcjach jądrowych (ang. kernel functions), który wykrywa odpowiednie zdarzenia w zebranych danych. Algorytm ten pozwala wyodrębnić zdarzenia elementarne z zapisanych obserwacji. Ponadto proponowany jest zarys systemu do automatycznego oddzielenia pomiarów kontrolnych i tych dokonanych w stresujących warunkach. Jako że małże z gatunku *Dreissena polymorpha* są znanymi wskaźnikami biologicznymi, jest to istotny krok w kierunku biologicznych systemów wczesnego ostrzegania.

2 Opis wyników rozprawy

2.1 Uwagi na temat architektury bazy danych dla szeregów czasowych



Rysunek 1: Ogólna architektura bazy danych dla szeregów czasowych

W rozprawie rozważane są problemy związane z bazami danych dla szeregów czasowych z obsługą GPGPU. Typowa architektura bazy danych dla szeregów czasowych składa się z trzech warstw: warstwy przechowywania danych, modułu wprowadzania danych oraz silnika zapytań. Założenia dotyczące architektury opisane poniżej (porównaj z Rysunkiem 1) były punktem wyjścia dla prowadzonych badań.

Aby osiągnąć wysoką wydajność i skalowalność, jako magazyn bazy danych może być wykorzystana jedna z baz danych NoSQL zgodnych z podejściem *Big Table*. Znane są przykłady wykorzystania w tym kontekście baz HBase i Cassandra [7, 13].

Kolektory danych są to elementy odpowiedzialne za gromadzenie oraz wprowadzanie do bazy danych obserwacji pochodzący z różnych źródeł. Proces ten musi uwzględniać fakt, że źródła danych ciągle generują nowe wartości. Zazwyczaj dane pochodzą z prowadzonych pomiarów lub monitorowania systemu operacyjnego, baz danych, serwerów sieciowych lub urządzeń sieciowych (zobacz opis narzędzi do monitorowania OpenTSDB [13]). System nie powinien w żaden sposób ograniczać zbieranych danych, optymalnie dane powinny być przechowywane bezstratnie. W większości zastosowań bezpiecznie jest założyć, że przetwarzane szeregi czasowe nie są równomiernie próbkowane. Szeregi czasowe są zapisywane jako pary *znacznik czasu* oraz *wartości pomiaru* (t_i, v_i). Każdy szereg czasowy powinien być jednoznacznie identyfikowalny poprzez metadane.

Można założyć że rozproszone kolektory danych (dopuszcza się wiele instancji, w tym takie które pracują bezpośrednio na źródle danych) dostarczają dane bezpośrednio do składu danych. Aby zapewnić optymalne przechowywanie danych *zarządca* powinien być używany do organizacji i kompresji danych. Ponadto, warto przyjąć, że szeregi czasowe są podzielone na rozłączne fragmenty i przechowywane w oddzielnych rekordach.

Użytkownicy mogą wykonywać zapytania przy użyciu dowolnej instancji *planera zapytań*, efektem czego będzie stworzenie rozproszonego planu zapytań. Stworzony plan zapytań może być przetwarzany przy wykorzystaniu heterogenicznych procesorów GPU/CPU. Planer zapytań powinien uwzględniać obciążenie poszczególnych jednostek w środowisku. Wygenerowany plan zapytania musi uwzględniać wiele czynników: szybkość CPU i GPU, dostępną pamięć, przepustowość pamięci, rozmiar danych, stopień kompresji i szybkość połączenia sieciowego. Ze względu na wymienione czynniki, nie wszystkie operacje mogą zostać przyspieszone poprzez wykorzystanie GPU. Dlatego zapytania powinny być wykonywane na GPU tylko wtedy, kiedy transfer danych z CPU na GPU jest amortyzowany przez szybsze przetwarzanie danych [4].

Jako że przetwarzanie zapytań w architekturze typu klient-serwer może przynieść wymierne korzyści [10], warto uwzględnić przypadek, w którym wykorzystywane są również zasoby sprzętowe po stronie klienta.

2.2 Metody lekkiej kompresji dla CPU/GPU

Wyniki badań dotyczących stosowania algorytmów lekkiej kompresji w obliczeniach typu GPGPU są opisane w publikacjach [15–17].

Zastosowanie metod lekkiej kompresji w systemach baz danych może znacząco poprawić ich wydajność [24, 25]. Systemy baz danych dla szeregów czasowych nie są wyjątkiem. Metody lekkiej kompresji mogą w istotny sposób poprawić wydajność takich systemów. Związane jest to z tym, że metody te mają duży wpływ na rozmiar oraz wydajność składu danych systemu. Algorytmy lekkiej kompresji projektowane są w taki sposób, aby przepustowość dekompresji była większa od przepustowości transferu danych z nośnika danych. Wzrost wydajności systemu bazodanowego jest więc związany ze wzrostem przepustowości danych.

Ponadto, metody lekkiej kompresji mogą poprawić wydajność obliczeń GPGPU w zastosowaniach bazodanowych [9, 15–17]. Dzieje się tak ze względu na mniejszy narzut na transmisję danych ze składu danych do pamięci RAM i dalej z pamięci RAM do globalnej pamięci urządzenia. Ponadto, w niektórych zastosowaniach, można dekompresować dane w locie, tylko na czas obliczeń. Dzięki temu można także poprawić czas dostępu do danych wewnątrz karty GPU. Takie podejście może być wykorzystane również w ogólnym kontekście obliczeń GPGPU, a w szczególności w problemach intensywnego przetwarzania danych.

Algorytmy lekkiej kompresji koncentrują się nie tylko na rozmiarze skompresowanych danych, ale są projektowane w taki sposób, aby w pierwszej kolejności poprawić przepustowość dekompresowanych danych. W związku z tym, warto rozważyć kaskadowe plany lekkiej kompresji, które wykorzystują wiele algorytmów lekkiej kompresji w celu poprawienia współczynnika kompresji [9, 17]. Kaskadowe plany kompresji są bardziej wymagające obliczeniowo, co z kolei może zredukować przepustowość dekompresji danych. Dlatego ważne jest, aby znaleźć równowagę pomiędzy stopniem kompresji a przepustowością dekompresji.

Ze względu na różne pochodzenie danych, szeregi czasowe różnią się swoją charakterystyką. Również różne fragmenty tego samego szeregu czasowego mogą mieć zróżnicowaną charakterystykę. Dlatego planer kompresji powinien być w stanie stosować różne plany kompresji dla różnych fragmentów szeregów czasowych [17].

Reasumując, w pracy [17] udało nam się poprawić wyniki wcześniej przedstawione przez Fang i innych w [9] poprzez:

- zaproponowanie nowych efektywnych implementacji trzech algorytmów lekkiej kompresji wykorzystujących mechanizm „Patch" danych dla platformy GPU,
- poprawienie niektórych z wcześniej prezentowanych implementacji algorytmów (np. poprzez dopuszczenie kodowania bitowego o dowolnej długości z przedziału $[2, \ldots, 32]$ – w przeciwieństwie do pracy [9], gdzie algorytmy dopuszczały tylko kodowania wykorzystujące wielokrotności bajtu).

Ponadto zaprezentowaliśmy koncepcję dynamicznego planera lekkiej kompresji dla szeregów czasowych [15], adaptującego się do zmiennych właściwości szeregów czasowych. Następnie wykazaliśmy, że lekka kompresja może znacznie poprawić wykorzystanie GPGPU przy obliczeniach intensywnych ze względu na dane, a w szczególności w zastosowaniach związanych z przetwarzaniem szeregów czasowych [15–17]. Wreszcie, udało się pokazać, że w pewnych zastosowaniach możliwe jest poprawienie czasu dostępu do pamięci wewnątrz GPU poprzez wykorzystanie dekompresji w locie.

2.3 Przetwarzanie zapytań dla szeregów czasowych z użyciem CPU/GPU

Tematyka przetwarzania szeregów czasowych jest omawiana w następujących publikacjach [15, 16, 18, 19].

Zagadnienie podobieństwa pomiędzy szeregami czasowymi jest ważnym aspektem szerszego problemu z zakresu eksploracji danych szeregów czasowych. Wiele z zagadnień

eksploracji szeregów czasowych opiera się o pojęcie podobieństwa szeregów czasowych. Przykładowo, zapytania typu: „query by content", „k-nearest neigbors" czy „ϵ-*range*", są to znane zagadnienia w dziedzinie eksploracji danych szeregów czasowych, których działanie uzależnione jest od miary podobieństwa szeregów czasowych [8]. W publikacji [15] omówiony został problem dopasowania całych szeregów poprzez obliczenie odległości Hamminga jako funkcji podobieństwa z wykorzystaniem obliczeń na GPU. Pomimo że sama implementacja funkcji odległości Hamminga na GPU jest wydajna obliczeniowo, transfer danych z pamięci RAM do pamięci na GPU jest głównym problemem i znacząco redukuje wydajność GPU w takich zastosowaniach. Z tego powodu konieczne było zastosowanie metod lekkiej kompresji (zobacz paragraf 2.2), aby wyeliminować wąskie gardło, którym jest transfer danych [15]. Z powodzeniem przeprowadzone zostały eksperymenty z innymi miarami podobieństwa dla szeregów czasowych, w szczególności z odległością *euklidesową* oraz metodą *Dynamic Time Warping (DTW)* [14, 21]. W obu przypadkach udało się zmniejszyć narzut związany z transferem danych. Jednakże w przypadku DTW problem był na tyle złożony obliczeniowo, że poprawa transferu danych nie wpłynęła znacząco na wydajność całości.

W pracy [17] omówiono implementacje na GPU efektywnych algorytmów przetwarzania i transformacji szeregów czasowych, opartych na operacjach realizowanych w OpenTSDB [13]. W artykule przedstawiono algorytmy redukcji, interpolacji i agregacji danych, czyli powszechnie realizowane zadania w wielu bazach danych dla szeregów czasowych. Podobnie jak w poprzednim przypadku, w celu pełnego wykorzystania potencjału GPU w obliczeniach, należało rozwiązać problem transferu danych. Raz jeszcze wykorzystanie algorytmów lekkiej kompresji okazało się skuteczne.

Pomimo tego, że badania przedstawione w publikacjach [18, 19] koncentrują się głównie na zagadnieniach związanych z systemem biomonitoringu, warto zauważyć, że system w dużym stopniu wymaga wsparcia ze strony bazy danych dla szeregów czasowych. W systemie wykorzystywane są różne techniki eksploracji danych, a w szczególności filtrowanie danych, dopasowywanie wzorców, ekstrakcja cech czy klasyfikacja szeregów czasowych. Pomimo że metody zastosowane w systemie monitoringu biologicznego nie zostały dotychczas zrealizowane na GPU, to dla niektórych z wykorzystywanych operacji takie implementacje istnieją. W szczególności, implementacja dyskretnej transformaty falkowej na GPU została przedstawiona w [23], natomiast wyszukiwanie oparte na k-NN na GPU było omawiane w [11, 12].

Wyniki tych badań pozwolą w przyszłości na stworzenie zaawansowanego systemu baz danych dla szeregów czasowych ze wsparciem GPU.

2.4 Model dwucelowy optymalizacji planów zapytań w heterogenicznych środowiskach CPU/GPU

Badania nad optymalizacją zapytań w heterogenicznych środowiskach CPU/GPU zostały omówione w publikacji [20].

Dotychczasowe badania wskazują, że wiele zadań systemu bazodanowego może być przyspieszone poprzez wykorzystanie obliczeń na kartach GPU [1–6, 15–17, 22]. Jednakże, należy wcześniej rozwiązać kilka problemów z tym związanych. W rozprawie

zostały już omówione metody lekkiej kompresji pozwalające poprawić przepustowość obowiązkowego transferu danych z CPU do GPU (zobacz rozdział 2.2). Nie rozwiązuje to jednak wszystkich problemów. Jednostki GPU są głównie zoptymalizowane pod kątem obliczeń numerycznych, dlatego tylko wybrane operacje bazodanowe mogą w pełni wykorzystać GPU. Ponadto, niewielkie zestawy danych są również problemem. W przypadku zbyt małych danych, wzrost wydajności może być nieznaczny lub wręcz ujemny. Dlatego ważne jest, aby wykonywać obliczenia zarówno na CPU jak i na GPU [20]. W rozprawie rozważane są obliczenia w heterogenicznym środowisku CPU/GPU. Omawiana jest metoda optymalizacji poszukująca równowagi między tymi dwoma platformami. Metoda opiera się na heurystycznym poszukiwaniu planów wykonania optymalnych w sensie Pareto ze względu na dwa cele z uwzględnieniem wielu jednoczesnych zapytań.

Planer zapytań może korzystać z jednego z trzech obsługiwanych typów optymalizacji:

- czas (minimalizacja czasu przetwarzania),
- koszt (minimalizacja kosztów przetwarzania),
- oba cele (minimalizacja łącząca czas i koszt).

Optymalizacja względem kosztu jest oparta o model finansowy wzorowany na rynku surowców, gdzie urządzenia są traktowane jako producenci a zapytania są interpretowane jako konsumenci zasobów. Koszt wykorzystania zasobów jest kontrolowany przez prawa popytu i podaży. Wyniki eksperymentalne są bardzo obiecujące. Uzyskano dobry poziom równoważenia obciążenia w połączeniu z lepszymi wynikami optymalizacji niż w innych pracach [20]. Co więcej, omawiane rozwiązanie oferuje mniejszą złożoność obliczeniową w porównaniu do innych metod wykorzystywanych w bazach danych ze wsparciem GPU.

2.5 Przykładowe zastosowania

W pracach na temat systemu biomonitoringu opisane zostały niektóre zastosowania baz danych dla szeregów czasowych. Publikacja [19] stanowi rozszerzenie wyników badań wcześniej przedstawionych w [18].

Biomonitoring skażenia wody jest jednym z ważnych aspektów ochrony zdrowia i środowiska. Wiele istniejących systemów monitorowania bada wodę tylko w wąskim zakresie substancji i bez ciągłej kontroli pracy. Z tego powodu, systemy oparte o żywe organizmy budzą coraz większe zainteresowanie i zyskują coraz większą popularność.

Niektóre gatunki małży są znanymi bioindykatorami i mogą być użyte przy tworzeniu biologicznych systemów wczesnego ostrzegania. Takie systemy używają długoterminowych obserwacji aktywności bioindykatora w celu monitorowania środowiska. W tych badaniach, analizowane są obserwacje zachowań małży z gatunku Dreissena polymorpha w celu wykrycia ewentualnych zagrożeń. Ponieważ obserwacje zachowań małży są wykonywane poprzez pomiar rozwarcia między muszlami w czasie, w naturalny sposób można wykorzystać system bazy danych dla szeregów czasowych. Opisane podejście umożliwia automatyczną ekstrakcję zachowań, rozróżnienie pomiędzy warunkami kontrolnymi i wywołującymi stres, a nawet rozróżnienie pomiędzy różnymi substancjami stresującymi. W tym celu wykorzystywane są różne techniki eksploracji i przetwarzania danych, takie jak

transformata falkowa, filtry, czy metody poszukiwania wzorców, a także ekstrakcja cech i klasyfikacja fragmentów szeregów czasowych.

2.6 Konkluzje

W tej rozprawie doktorskiej przedstawione zostało zagadnienie optymalizacji zapytań w heterogenicznym środowisku CPU/GPU w kontekście baz danych dla szeregów czasowych.

Pokazano, że poprzez użycie kart GPU jako koprocesora w bazie danych dla szeregów czasowych, można znacząco poprawić wydajność przetwarzania zapytań [15, 17]. Aby uzyskać wzrost wydajności należy jednak rozwiązać kilka istotnych kwestii związanych z przetwarzaniem na kartach GPU.

Jednym z problemów jest czasochłonny transfer danych pomiędzy CPU i GPU, który poprzedza większość obliczeń GPU. W rozprawie rozważane jest wykorzystanie metod lekkiej kompresji w celu zminimalizowania czasu potrzebnego na transfer danych, dzięki czemu można poprawić wydajność przetwarzanie danych. Omawiane są algorytmy lekkiej kompresji, szczegóły ich wykonania oraz możliwe zastosowania w kontekście GPGPU [15–17]. Przedstawione są zarówno nowe implementacje jak i rozszerzenia niektórych istniejących algorytmów na platformie GPU. Ponadto przedstawiona jest koncepcja adaptacyjnego planera lekkiej kompresji danych [17]. Uzyskane współczynniki kompresji proponowanego rozwiązania, jak również wysoka przepustowość dekompresji na GPU, są atrakcyjnym rozwiązaniem dla baz danych ze wsparciem ze strony GPU.

Kolejną rozważaną kwestią związaną z wykorzystaniem GPU jako koprocesora w środowisku bazy danych jest odpowiedni dobór zadań bazodanowych wykonywanych na GPU. Nie wszystkie zadania dobrze wpisują się w model obliczeń GPU. Może to być związane z rozmiarem lub typem zadania. W rozprawie omawiany jest model optymalizacji obliczeń w heterogenicznych środowiskach CPU/GPU [20]. Opisane podejście bazuje na modelu ekonomicznym, który naśladuje rynki towarowe i wykorzystuje optymalizację dwucelową, która łączy cele kosztu i czasu wykonania zapytania. Proponowana jest heurystyczna metoda poszukiwania optymalnych planów zapytań. Wstępne wyniki są bardzo obiecujące i pozwalają uzyskać zrównoważone obciążenie symulowanych urządzeń w połączeniu z lepszymi wynikami optymalizacji niż w wcześniejszych metodach [20].

Wreszcie omawiane są badania związane z monitorowaniem eksperymentu biologicznego. Jest to przykład zastosowań dla baz danych dla szeregów czasowych. W pracy analizowane są obserwacje zmiany rozwarcia między muszlami racicznicy zmiennej. Obserwacje te są oczywiście zbierane w postaci szeregów czasowych. Proponowany jest nowy algorytm oparty na transformacie falkowej i metodach jądrowych, który dokonuje ekstrakcji zachowań poprzez analizę szeregu czasowego obserwacji [18, 19]. Automatyczna ekstrakcja zdarzeń znacząco ułatwia dalsze badania nad zachowaniem małży z gatunku racicznica zmienna. Ponadto, przedstawione są wyniki wskazujące, że zastosowania tego algorytmu mogą być znacznie szersze. Algorytm może zostać wykorzystany w procesie automatycznej klasyfikacji zachowań małży w zależności od stanu otaczającego środowiska [19]. Jest to ważny etap na drodze ku stworzeniu zaawansowanego systemu

biomonitoringu środowiska w oparciu o ten gatunek małży.

2.7 Potencjalne kierunki kontynuacji badań

W przyszłości planowany jest dalszy rozwój metod przedstawionych w tej rozprawie doktorskiej.

W kontekście baz danych dla szeregów czasowych planowane jest rozszerzenie metod przetwarzania zapytań przy użyciu GPU, a w szczególności przetwarzanie zaawansowanych zapytań i eksploracji danych szeregów czasowych. Rozważana jest także adaptacja proponowanych metod na potrzeby systemów monitoringu przemysłowego typu SCADA.

Planowane są również szersze badania na temat zastosowania algorytmów lekkiej kompresji w bardziej ogólnym kontekście obliczeń GPGPU. W szczególności interesująca jest możliwość wykorzystania tych metod w kontekście różnych architektur wspierających model obliczeń SIMD. Planowana jest również kontynuacja badań na temat dynamicznego planera kompresji, w szczególności badania nad przetwarzaniem częściowo zdekompresowanych danych oraz wsparcie dla innych struktur danych (na przykład dla grafów).

Ponadto planowana jest szczegółowa ocena proponowanego modelu przetwarzania w heterogenicznych środowiskach CPU/GPU, a zwłaszcza badanie wpływu parametrów na zachowanie modelu oraz ocena modelu w kontekście wyzwań Hybrid Query opisanych w pracach [1–6]. Rozważane jest też rozszerzenie tego modelu poza tematykę koprocesorów.

Wreszcie, planowane są dalsze badania nad metodami analizy zachowań związanych z małżami z gatunku racicznica zmienna, z celem opracowania zaawansowanego systemu monitoringu biologicznego wody.

Literatura

[1] Breß, S. and Saake, G. Why it is time for a HyPE: A hybrid query processing engine for efficient GPU coprocessing in DBMS. *Proceedings of the VLDB Endowment*, 6(12):1398–1403, 2013.

[2] Breß, S., Beier, F., Rauhe, H., Schallehn, E., Sattler, K.-U., and Saake, G. Automatic selection of processing units for coprocessing in databases. In *Advances in Databases and Information Systems*, pages 57–70. Springer, 2012.

[3] Breß, S., Mohammad, S., and Schallehn, E. Self-tuning distribution of DB-operations on hybrid CPU/GPU platforms. *Grundlagen von Datenbanken, CEUR-WS*, pages 89–94, 2012.

[4] Breß, S., Geist, I., Schallehn, E., Mory, M., and Saake, G. A framework for cost based optimization of hybrid CPU/GPU query plans in database systems. *Control and Cybernetics*, pages 27–35, 2013.

[5] Breß, S., Heimel, M., Siegmund, N., Bellatreche, L., and Saake, G. Exploring the design space of a GPU-aware database architecture. *New Trends in Databases and Information Systems*, pages 225–234, 2013.

[6] Breß, S., Schallehn, E., and Geist, I. Towards optimization of hybrid CPU/GPU query plans in database systems. In *New Trends in Databases and Information Systems*, pages 27–35. Springer, 2013.

[7] Cloudkick. 4 Months with Cassandra, a love story, March 2010. URL `https://www.cloudkick.com/blog/2010/mar/02/4_months_with_cassandra/`. Archived copy `http://goo.gl/4rw3sW`.

[8] Esling, P. and Agon, C. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[9] Fang, W., He, B., and Luo, Q. Database compression on graphics processors. *Proceedings of the VLDB Endowment*, 3(1-2):670–680, 2010.

[10] Franklin, M. J., Jónsson, B. T., and Kossmann, D. Performance tradeoffs for client-server query processing. In *ACM SIGMOD Record*, volume 25, pages 149–160. ACM, 1996.

[11] Garcia, V., Debreuve, E., Nielsen, F., and Barlaud, M. K-nearest neighbor search: Fast GPU-based implementations and application to high-dimensional feature matching. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 3757–3760. IEEE, 2010.

[12] Garcia, V., Debreuve, E., and Barlaud, M. Fast k nearest neighbor search using GPU. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–6. IEEE, 2008.

[13] OpenTSDB. What's OpenTSDB, 2010-2014. URL http://opentsdb.net/.

[14] Poli, G., Levada, A., Mari, J., and Saito, J. Voice command recognition with dynamic time warping (dtw) using graphics processing units (gpu) with compute unified device architecture (cuda). In *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on*, pages 19–25. IEEE, 2007.

[15] Przymus, P. and Kaczmarski, K. Improving efficiency of data intensive applications on GPU using lightweight compression. In *On the Move to Meaningful Internet Systems: OTM 2012 Workshops, Lecture Notes in Computer Science*, volume 7567, pages 3–12. Springer Berlin Heidelberg, 2012.

[16] Przymus, P. and Kaczmarski, K. Time series queries processing with GPU support. In *New Trends in Databases and Information Systems*. 17th East-European Conference on Advances in Databases and Information Systems September 1–4, 2013, Genoa, Italy, 2013.

[17] Przymus, P. and Kaczmarski, K. Dynamic compression strategy for time series database using GPU. In *New Trends in Databases and Information Systems*. 17th East-European Conference on Advances in Databases and Information Systems September 1–4, 2013, Genoa, Italy, 2013.

[18] Przymus, P., Rykaczewski, K., and Wiśniewski, R. Application of wavelets and kernel methods to detection and extraction of behaviours of freshwater mussels. *Future Generation Information Technology*, pages 43–54, 2011.

[19] Przymus, P., Rykaczewski, K., and Wiśniewski, R. Zebra mussels' behaviour detection, extraction and classification using wavelets and kernel methods. *Future Generation Computer Systems*, 33 (0):81–89, 2014. Special Section on Applications of Intelligent Data and Knowledge Processing Technologies. Guest Editor: Dominik Ślęzak.

[20] Przymus, P., Kaczmarski, K., and Stencel, K. A bi-objective optimization framework for heterogeneous CPU/GPU query plans. In *CS&P 2013 Concurrency, Specification and Programming*. Proceedings of the 22nd International Workshop on Concurrency, Specification and Programming, September 25–27, 2013, Warsaw, Poland.

[21] Sart, D., Mueen, A., Najjar, W., Keogh, E., and Niennattrakul, V. Accelerating dynamic time warping subsequence search with GPUs and FPGAs. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1001–1006. IEEE, 2010.

[22] Walkowiak, S., Wawruch, K., Nowotka, M., Ligowski, L., and Rudnicki, W. Exploring utilisation of GPU for database applications. *Procedia Computer Science*, 1(1):505–513, 2010.

[23] Wong, T. T. and Heng, P. A. Discrete wavelet transform on GPU. 2004.

[24] Zukowski, M., Boncz, P., Nes, N., and Héman, S. MonetDB/X100–a DBMS in the CPU cache. *IEEE Data Eng. Bull*, 28(2):17–22, 2005.

[25] Zukowski, M., Heman, S., Nes, N., and Boncz, P. Super-scalar RAM-CPU cache compression. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 59–59. IEEE, 2006.